ELSEVIER

Theoretical Computer Science 183 (1997) 157–185

Theoretical Computer Science

Order-sorted algebraic specifications with higher-order functions[1]

Anne Elisabeth Haxthausen *

Department of Information Technology, Technical University of Denmark, bldg. 344, DK-2800 Lyngby, Denmark

Abstract

This paper gives a proposal for how order-sorted algebraic specification languages can be extended with higher-order functions. The approach taken is a generalisation to the order-sorted case of an approach given by Möller, Tarlecki and Wirsing for the many-sorted case. The main idea in the proposal is to only consider reachable extensional algebras. This leads to a very simple theory, where it is possible to relate the higher-order specifications to first-order specifications.

Keywords: Order-sorted algebraic specification; Higher-order functions; Subtypes

1. Introduction

The objective of this paper is to investigate how order-sorted algebraic specification languages can be extended with higher-order functions. Below we describe our goals and their background and give a summary of the contents of the paper.

1.1. Background

During the last decades many languages for specifying data types and functions have been researched and developed. Two major approaches may be distinguished: (1) the model-oriented and (2) the algebraic. The algebraic specification languages have the advantage that they allow a high abstraction level where one can abstract away from implementation details like data-type representations. However, in contrast to model-oriented specification languages and functional programming languages the algebraic specification languages (with a few exceptions like RSL [16]) do not allow

* E-mail: ah@it.dtu.dk.

[1] The work described in this paper was initiated during a visit at Electrotechnical Laboratory in Tsukuba in Japan and was supported by Japan International Science and Technology Exchange Center. The further development of the theory and the production of the paper has been supported by the Danish Technical Research Council under the "Codesign" programme.

PII S0304-3975(96)00323-4

higher-order functions. This is a pity, since higher-order functions are useful, for instance, for describing schematic algorithms such as generic tree-walking operations or for describing programming language semantics. Goguen shows in [3] how one can use parameterised specifications instead of using higher-order functions which take functions as parameters. However, it is simpler (shorter) to use higher-order functions, as these can directly be applied to actual parameter functions, while if parameterised specifications are used, one first has to define an actual parameter module providing the actual parameter function and then apply the parameterised module to this.

Therefore, we wish to extend algebraic specification languages with higher-order functions. Such extended languages we will refer to as *higher-order (algebraic) specification languages*. In particular, we wish to extend order-sorted algebraic specification languages like OBJ [2, 5] because they provide the possibility of specifying subtypes (subsorts) and thereby for defining certain partial functions as total functions on subtypes. Below is an example of what we may like to write:

sorts nat, nznat.
subsorts nznat $\leqslant$ nat.
ops
  zero : nat,
  succ : nat $\rightarrow$ nzat,
  twice : (nat $\rightarrow$ nat) $\rightarrow$ (nat $\rightarrow$ nat),
  succ2 : nat $\rightarrow$ nznat.
vars f : nat $\rightarrow$ nat, x : nat.
eqs
  twice(f)(x) = f(f(x)),
  succ2 = twice(succ).

Here *twice* is a higher-order function, which takes a function, f, as argument and returns another function, which when applied to some argument, x, returns the same as if f had been applied twice to x. The function *succ*2 is defined as the application of *twice* to *succ*.

Research has been done on how to give semantics to higher-order algebraic specification languages. One way is to use Cartesian closed categories as the mathematical foundations, as e.g. in [14] (for the many-sorted case) and [9] (for the order-sorted case). Another way is to extend the usual set-theoretical algebraic framework as e.g. in [11, 10] (for the many-sorted case) and [15] (for the order-sorted case). In [12, 13], Möller et al. do this implicitly for the many-sorted case by a transformational semantics: under the assumption that one is only interested in term-generated algebras, the semantics of a higher-order specification can be given by the semantics of a corresponding first-order specification. This transformational approach is very attractive because, in contrast to the other approaches, many definitions and theorems can directly be derived from the first-order case.

Therefore, one of the main goals of this paper is to investigate how the Möller et al. approach can be generalised to the order-sorted case.

1.2. Contents of paper

This paper gives, in two steps, in Sections 2 and 3, a proposal for higher-order order-sorted algebraic specification.

In Section 2, we keep the subtype relation so simple that it is possible to directly relate the higher-order order-sorted specifications to first-order order-sorted specifications, where the subtype-as-inclusion principle is used. In Section 3, we use a more general subtype relation for which the function-type constructor is anti-monotonic in its first argument, such that, for instance, a function of type $Int \to Int$ can be passed as an actual parameter to a (higher-order) function which requires a function of type $Nat \to Int$. For this more general relation, it is not possible to directly relate the higher-order order-sorted specifications to usual first-order order-sorted specifications – it is necessary to relate them to a notion of first-order generalised order-sorted specifications, which includes two kinds of subtypes: subtype-as-inclusion and subtype-as-implicit-coercion. We have developed such a notion in Appendix D. In [9] Martí-Oliet and Meseguer also include and distinguish the two kinds of subtypes in higher-order specifications; however their semantics is not based on a transformation to first-order specifications, but on Cartesian closed categories. Qian [15] also distinguishes two kinds of subtypes, but the second of these is less general than ours and is defined in a framework which is quite different from the usual algebraic framework.

Finally, in Section 4, a summary and discussion of the proposal is given.

For the convenience of the reader, Appendices A and B contain some well-known definitions and results from (first-order) many-sorted algebra (as defined in [1]) and (first-order) order-sorted algebra (as defined in [4]) on which other definitions in this paper depend. Furthermore, these appendices show which notation is used.

Appendix C contains (our own) definitions and results for the fundamentals of (first-order) algebraic specification with subtypes as implicit coercions. This is used in our definition of generalised algebra in Appendix D.

The paper extends our previous paper [7] by a further development of the theory for generalised higher-order order-sorted specifications in Section 3 and the underlying theory in Appendices C and D.

2. Higher-order order-sorted specification

When defining a notion of higher-order specifications, there are a number of tasks to be done. The notions of signatures, axioms, algebras, homomorphism and satisfaction relation should be decided. Furthermore, if initial algebra semantics should be used, the existence of initial algebras for specifications should be investigated. These tasks are done in the following.

The approach we take is a generalisation of the Möller et al. approach [12] for the many-sorted case. The main idea in this approach is, that under the assumption that we are only interested in reachable (term-generated) extensional higher-order algebras, we

can consider higher-order specifications as first-order specifications with implicit higher-order sorts, apply-functions and extensionality axioms. In this way we can easily derive a number of definitions and theorems from the first-order case.

2.1. Signatures

A higher-order order-sorted signature is like a first-order order-sorted signature (see Appendix B), but the sorts that may be used in the definitions of operation symbols are higher-order sorts, i.e. not just basic sorts, but also functional sorts as well. Functional sorts, $s1 \to s2$, are built by applying a built-in sort constructor, $\to$, to sorts $s1$ and $s2$ which may be basic or functional. Operation symbols having functional sorts are intended to behave like functions. Hence, in a higher-order specification, one can specify functions that take functions as argument and/or return functions. Below we give the precise definitions.

Definition 2.1. A *higher-order order-sorted signature HSIG* is a triple $(S, \leqslant, OP)$, where

(i) $(S, \leqslant)$ is a poset,

(ii) OP is a family $(OP_s)_{s \in S^{\to}}$ of distinct $S^{\to}$-sorted constant symbols.

The set $S^{\to}$ of *higher-order sorts* generated from S is the least set for which:

(i) $S \subseteq S^{\to}$,

(ii) $w \to s \in S^{\to}$, if $w \in (S^{\to})^{+}$, $s \in S^{\to}$.

Note that compared with first-order order-sorted signatures, we only allow constant symbols – we do not need operation symbols of the form $\sigma : w \mapsto s$, as we can now define constants having functional sorts, i.e. we have constant symbols of the form $\sigma : w \to s$. This restriction is solely made in order to avoid confusion between the two forms – there would be no theoretical problems in allowing both. As a consequence of the restriction and the fact that constant symbols cannot be overloaded, function symbols cannot be overloaded.

One of the choices we must make is how the subsort relation, $\leqslant$, on the basic sorts in S should be extended to a relation on the higher-order sorts in $S^{\to}$. In this section, we will use a relation, $\leqslant^{\to}$, for which $\to$ is constant in its first argument, and monotonic in its second argument. With this relation we shall see that it is possible to relate our higher-order order-sorted specifications to first-order order-sorted specifications. In Section 3 we will consider a more general relation, $\leqslant^{\Rightarrow}$, for which $\to$ is anti-monotonic in its first argument, and monotonic in its second argument.

Definition 2.2. Let $(S, \leqslant)$ be a poset. Then $(S^{\to}, \leqslant^{\to})$ is a poset generated from $(S, \leqslant)$ with the ordering relation $\leqslant^{\to}$ being the least relation satisfying

(i) $s \leqslant^{\to} s'$, if $s \leqslant s'$,

(ii) $w \to s \leqslant^{\to} w \to s'$, if $s \leqslant^{\to} s'$.

Definition 2.3. The first-order order-sorted signature *associated* with a higher-order signature $HSIG = (S, \leqslant, OP)$ is

$$HSIG^{\rightarrow} = (S^{\rightarrow}, \leqslant^{\rightarrow}, OP^{\rightarrow}),$$

where $OP^{\rightarrow} = OP \cup (\{apply\}_{\langle w \rightarrow s, w\rangle, s})_{w \in (S^{\rightarrow})^+, s \in S^{\rightarrow}}$.

Notation. For terms (see next section) of the form $apply(f,x)$, one could use a more appealing notation like $f(x)$. This convention has been used in the example in the introduction.

Note that the constant symbols in *HSIG* must be distinct, which is a necessary and sufficient condition for ensuring that the monotonicity condition for $HSIG^{\rightarrow}$ is satisfied.

We are interested in signatures being regular and locally upward filtered, as regularity ensures that term algebras are initial, and locally upward filteredness ensures that equational satisfaction is closed under isomorphism. Below we define these properties and give some facts about when they hold.

Definition 2.4. A higher-order order-sorted signature *HSIG* is *regular/locally upward filtered/coherent* iff $HSIG^{\rightarrow}$ is regular/locally upward filtered/coherent. [1]

Proposition 2.5. *All higher-order order-sorted signatures are regular.*

Proof. Given a signature $HSIG = (S, \leqslant, OP)$, we shall prove that for any $\sigma : w \mapsto s$ in $OP^{\rightarrow}$ and $w0 \leqslant w$, there is a least rank $w1 \mapsto s1$ for σ for which $w0 \leqslant w1$. The only symbols in $OP^{\rightarrow}$ are constants and *apply* operations. The constant symbols obviously satisfy the requirement. Any *apply* operation will have a rank of the form $(w \rightarrow s)w \mapsto s$. Any lower bound for its arity will be of the form $(w \rightarrow s0)w0$, where $s0 \leqslant^{\rightarrow} s$ and $w0 \leqslant^{\rightarrow} w$. Now among the *apply* operations there is one with rank $(w \rightarrow s0)w \mapsto s0$, and this is obviously the least rank, for which the arity is an upper bound for $(w \rightarrow s0)w0$. □

Proposition 2.6. *A higher-order order-sorted signature $HSIG = (S, \leqslant, OP)$ is locally upward filtered, if the connected components of $(S, \leqslant)$ are locally upward filtered.*

Proof. Follows from the fact that the sort constructor $\rightarrow$ is monotone wrt. $\leqslant^{\rightarrow}$. □

Corollary 2.7. *A higher-order order-sorted signature $HSIG = (S, \leqslant, OP)$ is coherent, if the connected components of $(S, \leqslant)$ are locally upward filtered.*

[1] For a definition, see Appendix B.

2.2. Terms and axioms

Definition 2.8. Given a higher-order order-sorted signature *HSIG*. A *higher-order order-sorted HSIG-term/axiom* is a first-order order-sorted $HSIG^{\rightarrow}$-term/axiom.[2]

Proposition 2.9. *Any HSIG-term has a least sort.*

Proof. Follows from Fact 2.5 and B.11. □

2.3. Specifications

Definition 2.10. A *higher-order order-sorted specification HSPEC* is a pair (*HSIG*, *HE*) consisting of a higher-order order-sorted signature *HSIG* and a set *HE* of higher-order order-sorted *HSIG*-axioms.

2.4. Algebras, homomorphisms and satisfaction relation

As higher-order *HSIG*-algebras we will use extensional $HSIG^{\rightarrow}$-algebras. The extensionality ensures that any carrier $A_{w\rightarrow s}$ of a functional sort $w \rightarrow s$ is isomorphic to a subset of the function space $A_w \rightarrow A_s$.

Definition 2.11. Given a higher-order order-sorted signature *HSIG*. *A higher-order order-sorted HSIG-algebra*, *A*, is an extensional first-order order-sorted $HSIG^{\rightarrow}$-algebra.[3]

Definition 2.12. A $HSIG^{\rightarrow}$-algebra, *A*, is *extensional* if, for all sorts $w \rightarrow s \in S^{\rightarrow}$ and $f, g \in A_{w\rightarrow s}$ the following holds:

$$(\forall a \in A_w \bullet apply^A(f,a) = apply^A(g,a)) \Rightarrow f = g.$$

As higher-order order-sorted *HSIG*-algebras, terms and axioms are first-order order-sorted $HSIG^{\rightarrow}$-algebras, terms and axioms, the notions of *HSIG-homomorphisms*, *HSIG-evaluation* of terms, *HSIG-satisfaction* of *HSIG*-axioms by *HSIG*-algebras etc. carry directly over from the first-order case, and we will not bother the reader with these definitions.

Notation. The *HSIG*-algebras and *HSIG*-homomorphisms form a category denoted *HAlg*(*HSIG*). The reachable *HSIG*-algebras and the *HSIG*-homomorphisms between these form a category denoted *RHAlg*(*HSIG*).

It is possible to give a first-order order-sorted specification, such that its reachable algebras coincide with the reachable *HSIG*-algebras:

---

[2] First-order axioms are conditional equations, cf. Appendix B.

[3] For a definition of the notion of first-order order-sorted algebras, see Appendix B.

Theorem 2.13. *For any higher-order signature HSIG it holds that*

$$RHAlg(HSIG) = ROSAlg((HSIG^{\rightarrow}, ext(HSIG))$$

where ext(HSIG) consists of exactly one first-order order-sorted ground infinitary conditional equation

$$t = t' \quad \textbf{if} \bigwedge_{t'' \in T_w} (apply(t, t'') = apply(t', t''))$$

for each $t,\ t' \in T_{w \to s},\ w \in (S^{\rightarrow})^+,\ s \in S^{\rightarrow}$.

Proof. Follows from the fact that for any reachable $HSIG^{\rightarrow}$-algebra, A, $A_s = \{eval_s^A(t) \mid t \in T_s(HSIG^{\rightarrow})\}$, where $eval^A$ is the unique A evaluation homomorphism, and therefore A satisfies $ext(HSIG)$ if, and only if A is extensional. □

Fact 2.14. *In general, not all $HSIG^{\rightarrow}$-algebras satisfying ext(HSIG) are extensional, and vice versa; cf.* [12] *where this is shown for the many-sorted case.*

Definition 2.15. Given a higher-order order-sorted specification $HSPEC = (HSIG, HE)$. A *higher-order order-sorted HSPEC-algebra*, A, is a $HSIG$-algebra satisfying each of the axioms in HE (in other words an extensional $HSIG^{\rightarrow}$-algebra satisfying each of the axioms in HE).

Notation. The $HSPEC$-algebras and $HSPEC$-homomorphisms form a category denoted $HAlg(HSPEC)$. The category of reachable $HSPEC$-algebras and $HSPEC$-homomorphisms between these is denoted $RHAlg(HSPEC)$.

For each higher-order specification, $HSPEC$, it is possible to give a first-order order-sorted specification, $HSPEC^{\rightarrow}$, such that the reachable $HSPEC^{\rightarrow}$-algebras coincide with the reachable $HSPEC$-algebras:

Definition 2.16. The first-order order-sorted specification *associated* with a higher-order order-sorted specification $HSPEC = (HSIG, HE)$ is

$$HSPEC^{\rightarrow} = (HSIG^{\rightarrow}, HE \cup ext(HSIG)).$$

Theorem 2.17. *For any higher-order specification HSPEC it holds that*

$$RHAlg(HSPEC) = ROSAlg(HSPEC^{\rightarrow}).$$

Proof. Follows from Theorem 2.13 and Definition 2.15. □

2.5. Initiality theorems

Theorem 2.18. *Given a coherent higher-order order-sorted signature HSIG. Then there exists an initial algebra in RHAlg(HSIG). If furthermore HSIG is sensible*

(see Definition 2.19*), then the* $HSIG^{\rightarrow}$*-term algebra,* $T(HSIG^{\rightarrow})$*, is one of the initial algebras in* $RHAlg(HSIG)$*.*

Proof. First part follows from Theorem 2.13 and the fact that there exists an initial algebra in $ROSAlg(HSIG^{\rightarrow}, ext(HSIG))$, if $HSIG^{\rightarrow}$ is coherent. The second part follows from the fact that $T(HSIG^{\rightarrow})$ satisfies $ext(HSIG)$, if $HSIG$ is sensible. □

Definition 2.19. A higher-order signature, $HSIG=(S, \leqslant, OP)$, is *sensible*, if for all functional sorts $w \rightarrow s \in S^{\rightarrow}$ it holds that

$$T_{w\rightarrow s} \neq \{\} \Rightarrow (T_w \neq \{\} \vee \textbf{card}\ T_{w\rightarrow s} = 1).$$

Theorem 2.20. *Given a higher-order order-sorted specification* $HSPEC=(HSIG, HE)$ *with coherent signature, HSIG. Then there exists an initial algebra in* $RHAlg(HSPEC)$*. If furthermore, HSIG is sensible, then the* $HSPEC^{\rightarrow}$ *quotient ground term algebra,* $T(HSPEC^{\rightarrow})$*, is one of the initial algebras in* $RHAlg(HSIG)$*.*

Proof. The first part follows from Theorem 2.17 and the fact that there exists an initial algebra in $ROSAlg(HSPEC^{\rightarrow})$, if $HSIG^{\rightarrow}$ is coherent. The second part follows from the fact that $T(HSPEC^{\rightarrow})$ satisfies $ext(HSIG)$, if $HSIG$ is sensible. □

3. Generalised higher-order order-sorted specification

In this section, we wish to use the same approach as in last section, but for a subtype relation $\leqslant^{\Rightarrow}$ which is more general than $\leqslant^{\rightarrow}$. The following example gives a motivation for this.

Example 3.1. Consider the following specification:

sorts nat, int.
subsorts nat $\leqslant$ int.
ops
g : (nat $\rightarrow$ int) $\rightarrow$ int,
f : int $\rightarrow$ int.

It would be reasonable if $apply(g, f)$ was a term. However, this is only the case if $int \rightarrow int$ is a subtype of $nat \rightarrow int$. Unfortunately, that is not the case for the subtype relation $\leqslant^{\rightarrow}$, but for the subtype relation $\leqslant^{\Rightarrow}$, we define below.

Assume given a higher-order order-sorted signature $HSIG=(S, \leqslant, OP)$.

Definition 3.2. Let the relation $\leqslant^{\Rightarrow}$ on $S^{\rightarrow}$ be the least relation satisfying
(i) $s \leqslant^{\Rightarrow} s'$, if $s \leqslant s'$,
(ii) $s'_1 \ldots s'_n \rightarrow s \leqslant^{\Rightarrow} s_1 \ldots s_n \rightarrow s'$, if $s \leqslant^{\Rightarrow} s'$, and $s_i \leqslant^{\Rightarrow} s'_i$ for $i = 1, \ldots, n$.

Fact 3.3. *$\leqslant^{\rightarrow}$ is a subrelation of $\leqslant^{\Rightarrow}$.*

The generalised subtype relation ($\leqslant^{\Rightarrow}$) is the same as in [9], but more general than the one in [15], where, in the second rule above, $s_i \leqslant^{\Rightarrow} s_i'$ is replaced with $s_i \leqslant^{\rightarrow} s_i'$.

We now try to proceed in the same way as before by defining an associated first-order signature $HSIG^{\Rightarrow} = (S^{\rightarrow}, \leqslant^{\Rightarrow}, OP^{\rightarrow})$ for each higher-order signature $HSIG$, such that $RHAlg(HSIG) = RAlg(HSIG^{\Rightarrow}, ext(HSIG))$. However, this time we encounter a number of problems:

(i) the signatures are not regular,
(ii) the signatures are not always locally upward filtered,
(iii) subtypes as inclusion does not give all the desired models.

In the following, we will show how the two first problems can be solved by modifying the notions of order-sorted algebra, and the last by allowing subtypes as implicit coercions as well as subtypes as inclusion.

3.1. Problem: Signatures are not regular

We want $HSIG^{\Rightarrow}$ to be regular, otherwise there are terms which do not have a unique least parse.

Fact 3.4. *For any signature HSIG, $HSIG^{\Rightarrow}$ is not regular, but only pre-regular.* [4]

Example 3.5. Consider the following signature, *HSIG*:

sorts s, s1, s2.
subsorts s1 $\leqslant$ s2.
ops
f : s2 $\rightarrow$ s,
a : s1.

Consider the sort string, $w0 = (s2 \rightarrow s)s1$. The *apply* operations in $HSIG^{\rightarrow}$ with arities w, for which $w0 \leqslant^{\Rightarrow} w$, are:

(i) $apply : (s2 \rightarrow s)s2 \mapsto s$,
(ii) $apply : (s1 \rightarrow s)s1 \mapsto s$.

None of these have a least rank, as $s1 \leqslant^{\Rightarrow} s2$ and $s2 \rightarrow s \leqslant^{\Rightarrow} s1 \rightarrow s$. Therefore *HSIG* is not regular. This has the consequence that the term $apply(f, a)$ has two possible parses – none of which is "least":

(i) $apply : (s2 \rightarrow s)s2 \mapsto s, f : s2 \rightarrow s, a : s2$,
(ii) $apply : (s1 \rightarrow s)s1 \mapsto s, f : s1 \rightarrow s, a : s1$.

However, *HSIG* is pre-regular, since there is always a least co-arity. Hence, the term $apply(f, a)$ has a least sort: s.

[4] For a definition, see Appendix B.

The problem that there may be several possible parses is not serious, if we add the extra requirement to algebras, A, that, whenever there are more than one possible parse of a term, t, the meaning of t in A, using either of them, is the same. For the example above it means that

$$apply^A_{(s2\to s)s2,s}(f^A_{s2\to s}, a^A_{s1}) = apply^A_{(s1\to s)s1,s}(f^A_{s2\to s}, a^A_{s1}).$$

In general, it means that the problem can be solved if we change Definition B.8 to the following definition.

Definition 3.6. Given a pre-regular first-order order-sorted signature $SIG = (S, \leqslant, OP)$. A *modified first-order order-sorted SIG-algebra*, A, is a first-order many-sorted SIG-algebra satisfying the following condition:

$$\forall a \in A_{w0} \bullet \sigma^A_{w1,s1}(a) = \sigma^A_{w2,s2}(a), \quad \text{if } \sigma \in OP_{w1,s1} \cap OP_{w2,s2}, w0 \leqslant w1 \text{ and } w0 \leqslant w2$$

Note, this condition implies the usual monotonicity condition.

With this modified notion of algebras, it is sufficient in theorems about initiality etc. to require signatures to be pre-regular instead of regular, cf. [4].

3.2. Problem: Signatures are not always locally upward filtered

We are interested in when the connected components of $(S^{\to}, \leqslant^{\Rightarrow})$ are locally upward filtered, as this ensures that equational satisfaction is closed under isomorphism.

Fact 3.7. *That the connected components of $(S, \leqslant)$ are locally upward filtered does not imply that the connected components of $(S^{\to}, \leqslant^{\Rightarrow})$ are locally upward filtered. $(S^{\to}, \leqslant^{\Rightarrow})$ is locally upward filtered, if $(S, \leqslant)$ is locally upward as well as downward filtered.*

This is unfortunate, since in practice the connected components of $(S, \leqslant)$ are not always locally downward filtered.

Example 3.8. For $(S, \leqslant)$ defined by

sorts s1, s2, s3, s,
subsorts s1 ⩽ s3, s2 ⩽ s3,

it holds that $s3 \to s \leqslant^{\Rightarrow} s1 \to s$ and $s3 \to s \leqslant^{\Rightarrow} s2 \to s$, but $s1 \to s$ and $s2 \to s$ do not have an upper bound. Therefore not all connected components of $(S^{\to}, \leqslant^{\Rightarrow})$ are locally upward filtered.

This problem can be solved by changing the definition of the notion of axioms to the following.

Definition 3.9. Given a pre-regular first-order order-sorted signature $SIG = (S, \leqslant, OP)$. A *modified first-order order-sorted SIG-axiom* is a conditional equation of the form

$$(\forall X)t = t' \quad \textbf{if} \bigwedge_{i \in I} (t_i = t_i')$$

where I is a finite or infinite set of indices; $t, t', t_i, t_i' \in T(SIG, X)$ for all $i \in I$; there exists an upper bound of $LS(t)$ and $LS(t')$ in $(S, \leqslant)$, and for each $i \in I$, there exists an upper bound of $LS(t_i)$ and $LS(t_i')$ in $(S, \leqslant)$.

With this modified notion of axioms, equational satisfaction is closed under isomorphism, also for signatures that are not locally upward filtered.

3.3. Problem: Subtypes as inclusion not sufficient

If we use the subtype-as-inclusion principle, the algebras would not comprise all what we might expect.

Example 3.10. Assume $nat \leqslant int$, and thereby $int \to int \leqslant^{\Rightarrow} nat \to int$. Using the subtype-as-inclusion principle for the algebras A, we would get

$$A_{int \to int} \subseteq A_{nat \to int}$$

which would rule out algebras, where $A_{nat} = Nat$, $A_{int} = Int$, $A_{int \to int} \subseteq A_{int} \to A_{int}$ and $A_{nat \to int} \subseteq A_{nat} \to A_{int}$, since $A_{int} \to A_{int}$ and $A_{nat} \to A_{int}$ are disjoint.

This problem can be solved by using the subtype-as-implicit-coercion principle for subtypes. According to this principle, whenever $s \leqslant^{\Rightarrow} s'$ there must be a coercion function, $c^A_{s,s'}$, mapping values in A_s to values in $A_{s'}$. For the example above it means that $A_{int \to int}$ is not required to be a subset of $A_{nat \to int}$, but instead there must be a coercion function, $c^A_{int \to int, nat \to int}$, mapping functions, f, in $A_{int \to int}$ to functions, f', in $A_{nat \to int}$, in such a way that f and f' give the same results when applied to values in the A_{nat}. (In other words $c^A_{int \to int, nat \to int}(f)$ can be considered as the restriction of f to its subdomain A_{nat}.) The meaning in A of a term like $apply(g, f)$ from Example 3.1 should then be $apply^A_{((nat \to int) \to int)(nat \to int), int}(g^A, c^A_{int \to int, nat \to int}(f^A))$. Note that the coercion functions need not be injective, which implies that there may be loss of information, when coercing a value from a subtype to a supertype. For instance, for two different functions $f1$ and $f2$ in $A_{int \to int}$ returning the same result for natural numbers, we have $c^A_{int \to int, nat \to int}(f1) = c^A_{int \to int, nat \to int}(f2)$.

In Appendix C we have developed a first-order coercion algebra facilitating this view. However, we wish, like Martí-Oliet and Meseguer in [9], to include and distinguish between the two principles for subtypes. To be more precise, we wish to use the subtype-as-implicit-coercion principle for $\leqslant^{\Rightarrow}$, and the subtype-as-inclusion principle for $\leqslant^{\rightarrow}$. Therefore, it has been necessary to develop a first-order generalised algebra, which includes both order-sorted algebra and coercion algebra. This is presented in Appendix D.

3.4. The final solution

The final solution is to do exactly as in Section 2, but instead of using order-sorted algebra as the first-order framework, we use generalised algebra.

Let in the following be given a higher-order order-sorted signature $HSIG = (s, \leqslant, OP)$.

First we define $HSIG^{\Rightarrow}$:

Definition 3.11. The *associated* first-order generalised signature of $HSIG$ is

$$HSIG^{\Rightarrow} = (S^{\rightarrow}, \leqslant^{\rightarrow}, \leqslant^{\Rightarrow}, OP^{\rightarrow}).$$

Then the remaining definitions and theorems can be given as in Section 2, except that we use $HSIG^{\Rightarrow}$ instead of $HSIG^{\rightarrow}$, and "first-order generalised" instead of "first-order order-sorted".

Definition 3.12. A *generalised higher-order order-sorted HSIG-term/axiom* is a first-order generalised $HSIG^{\Rightarrow}$-term/axiom.

In other words, a $HSIG$-term/axiom is a first-order coercion $(S^{\rightarrow}, \leqslant^{\Rightarrow}, OP^{\rightarrow})$-term/axiom.

Note that in coercion axioms, one can indicate over which sort an equality should hold, by giving a sort subscript on the equality-operator. A motivation for this is given in Appendix C.4.

Definition 3.13. A *generalised higher-order order-sorted HSIG-algebra*, A, is an extensional first-order generalised $HSIG^{\Rightarrow}$-algebra.

In other words, a $HSIG$-algebra, A, is an extensional first-order coercion $(S^{\rightarrow}, \leqslant^{\Rightarrow}, OP^{\rightarrow})$-algebra, for which $A_s \subseteq A_{s'}$, and $c^A_{s,s'}$ is the identity function, for $s \leqslant^{\rightarrow} s'$.

Notation. The $HSIG$-algebras and $HSIG$-homomorphisms form a category denoted $HAlg(HSIG)$. The reachable $HSIG$-algebras and the $HSIG$-homomorphisms between these form a category denoted $RHAlg(HSIG)$.

It is possible to give a first-order generalised specification, such that its reachable algebras coincide with the reachable $HSIG$-algebras:

Theorem 3.14. $RHAlg(HSIG) = RGAlg((HSIG^{\Rightarrow}, ext(HSIG)))$, *where* $ext(HSIG)$ *consists of exactly one ground infinitary conditional equation*

$$t =_{w \to s} t' \quad \textbf{if} \bigwedge_{t'' \in T_w} (apply(t, t'') =_s apply(t', t''))$$

for each $t, t' \in T_{w \to s}$, $w \in (S^{\rightarrow})^+$, $s \in S^{\rightarrow}$.

Proof. Follows from the fact that for any reachable $HSIG^{\Rightarrow}$-algebra, $A, A_s = \{eval_s^A(t) \mid t \in T_s(HSIG^{\Rightarrow})\}$, and therefore A satisfies $ext(HSIG)$ if and only if A is extensional. □

Theorem 3.15. *There exists an initial algebra in RHAlg(HSIG).*

Proof. Follows from Theorems 3.14 and D.8 □

Example 3.16. Consider the following signature for binary numbers:

BIN =
sorts Zero, Bin.
subsorts Zero ⩽ Bin.
ops
zero : Zero,
one : Bin
f,g : Bin → Bin.

As $Zero \leqslant Bin$, we have $Bin \to Bin \leqslant^{\Rightarrow} Zero \to Bin$, and consequently f and g have sort $Bin \to Bin$ as well as sort $Zero \to Bin$. Therefore, we get two extensionality-axioms in $ext(BIN)$:

f $=_{Bin \to Bin}$ g **if**

(apply(f, zero) $=_{Bin}$ apply(g, zero)) ∧

(apply(f, one) $=_{Bin}$ apply(g, one)),

f $=_{Zero \to Bin}$ g **if** (apply(f, zero) $=_{Bin}$ apply(g, zero)).

An example of a reachable BIN-algebra (i.e. a reachable $BIN^{\Rightarrow}$-algebra satisfying the two extensionality axioms given above) is the algebra A defined as follows:

$$A_{Zero} = \{0\}, A_{Bin} = \{0, 1\}, A_{Bin \to Bin} = \{f^A, g^A\}, A_{Zero \to Bin} = \{h\},$$

$$zero^A = 0, one^A = 1,$$

$$f^A(0) = f^A(1) = 1 = g^A(0), g^A(1) = 0, h(0) = 1,$$

$$C^A_{Zero, Bin}(0) = 0,$$

$$c^A_{Bin \to Bin, Zero \to Bin}(f^A) = c^A_{Bin \to Bin, Zero \to Bin}(g^A) = h.$$

Empty carriers and coercion functions involving these are not shown.

The algebra satisfies the following BIN-axioms:

vars x : Bin.
eqns

$$\text{apply}(f, x) = \text{one},$$
$$\text{apply}(g, \text{zero}) = \text{one},$$
$$\text{apply}(g, \text{one}) = \text{zero}.$$

Note, that in these axioms, we could drop the sort subscript on the equality-operators, as there is only one upper bound (*Bin*) of the least sorts of the left- and right-hand sides.

4. Conclusions

4.1. Main results

In Sections 2 and 3, we gave proposals for the fundamentals of higher-order order-sorted algebraic specification. The approach we took was a generalisation of the Möller et al. approach in [12] from the many-sorted case to the order-sorted case.

The main idea in the approach is only to consider reachable extensional algebras. This leads to a very simple theory, where it is possible to relate the higher-order specifications to first-order specifications. To be more precise, a notion of higher-order specifications is defined, such that for each higher-order specification *HSPEC*, a first-order specification $HSPEC^{\rightarrow}$ can be derived, such that the class of reachable $HSPEC^{\rightarrow}$-algebras is equal to the class of reachable *HSPEC*-algebras.

One of the choices we had to make was how the subsort relation, $\leqslant$, on the basic sorts should be extended to a relation on the higher-order sorts. First, in Section 2, we tried the ideas out for a relation, $\leqslant^{\rightarrow}$, for which $\rightarrow$ was constant in its first argument, and monotonic in its second argument. In this case everything turned out smoothly. The main results were:

(i) definitions of syntactic and semantic notions,
(ii) existence of an initial reachable extensional algebra.

Then, in Section 3, we tried to do the same, but for a more general relation $\leqslant^{\Rightarrow}$, for which $\rightarrow$ was anti-monotonic in its first argument and monotonic in its second argument. In this case it turned out that we could not relate our higher-order order-sorted specifications to first-order order-sorted specifications. We therefore developed a new notion of first-order generalised algebra, which includes and distinguishes between the two principles for sub-types: subtype-as-inclusion and subtype-as-implicit-coercion. (This notion of first-order generalised algebra, may be seen as a result in itself). By relating the higher-order order-sorted specifications to first-order generalised specifications, instead of first-order order-sorted specifications, everything turned out smoothly.

4.2. Advantages and disadvantages

Some of the advantages of this approach are:

(i) the semantics is simple,

(ii) properties may easily be derived from the first-order case.
(iii) there is only one kind of function arrow.
In order to avoid two kinds of function arrows, we choose only to allow constant symbols in the signatures. The price of this is that function symbols (which are then constant symbols) must be distinct.

4.3. Topics for future work

Topics for future work include:
(i) to investigate if it is possible to allow constant symbols in first-order generalised signatures to be overloaded,
(ii) to invent a proof system for generalised algebra.

Appendix A. First-order many-sorted algebra

This appendix contains some well-known definitions from many-sorted algebra on which other definitions in this paper depend. Furthermore, it shows which notation we use. For a full treatment of the topic, we refer to [1].

Definition A.1. A *many-sorted signature* is a pair (S, OP), where
(i) S is a set of *sorts*
(ii) OP is a $S^* \times S$-sorted family $(OP_{w,s})_{w \in S^*, s \in S}$ of *operation symbols*.

Notation. We write $\sigma : w \mapsto s \in OP$ for $\sigma \in OP_{w,s}$; $\sigma : s$ for $\sigma : \mapsto s$; and OP_s for $OP_{w,s}$, if w is the empty string λ. For an operation symbol $\sigma : w \mapsto s$, we call $w \mapsto s$ or $\langle w, s \rangle$ for its *rank*, w for its *arity*, and s for its *co-arity*. Operation symbols having the empty string as arity are called *constant symbols*.

Definition A.2. Given a many-sorted signature $SIG = (S, OP)$. A *many-sorted SIG-algebra*, A, consists of
(i) a carrier set A_s for each $s \in S$,
(ii) a constant $\sigma_s^A \in A_s$ for each $\sigma \in OP_s$,
(iii) a function $\sigma_{w,s}^A : A_w \mapsto A_s$ for each $\sigma \in OP_{w,s}, w \neq \lambda$.

Notation. We write A_w for $A_{s1} \times \cdots \times A_{sn}$, when $w = s1 \ldots sn$. We write σ^A for σ_s^A and $\sigma_{w,s}^A$, when this does not give confusion.

Definition A.3. Given a many-sorted signature $SIG = (S, OP)$, and two SIG-algebras A and B. A *many-sorted SIG-homomorphism*, $h : A \mapsto B$, from A to B is an S-sorted family $(h_s)_{s \in S}$ of functions $h_s : A_s \mapsto B_s$ satisfying the following *homomorphism conditions*:
(i) $h_s(\sigma_s^A) = \sigma_s^B$, for constant symbols $\sigma \in OP_s$,
(ii) $\forall a \in A_w \bullet h_s(\sigma_{w,s}^A(a)) = \sigma_{w,s}^B(h_w(a))$, for non-constant symbols $\sigma \in OP_{w,s}$.

Notation. We write $h_w(a)$ for $(h_{s1}(a1),\ldots,h_{sn}(an))$, when $w=s1\ldots sn$ and $a=(a1,\ldots an)$.

Appendix B. First-order order-sorted algebra

This appendix contains some well-known definitions and results from order-sorted algebra on which other definitions in this paper depend. For a full treatment of the topic, we refer to [4].

B.1. Signatures

Definition B.1. An *order-sorted signature* is a triple $(S, \leqslant, OP)$ such that
(i) (S, OP) is a many-sorted signature,
(ii) $(S, \leqslant)$ is a poset (i.e. reflexive, transitive and anti-symmetric),
(iii) the following *monotonicity condition* is satisfied: if $\sigma \in OP_{w1,s1} \cap OP_{w2,s2}$ and $w1 \leqslant w2$ then $s1 \leqslant s2$.

Notation. We write $w \leqslant w'$ for $s1 \leqslant s1' \wedge \cdots \wedge sn \leqslant sn'$, when $w=s1\ldots sn$ and $w'=s1'\ldots sn'$.

Definition B.2. $SIG=(S, \leqslant, OP)$ is *regular* iff for any $\sigma \in OP_{w1,s1}$ and $w0 \leqslant w1$, there exists a least $\langle w,s\rangle$ for which $\sigma \in OP_{w,s}$ and $w0 \leqslant w$.

Definition B.3. $SIG=(S, \leqslant, OP)$ is *pre-regular* iff for any $\sigma \in OP_{w1,s1}$ and $w0 \leqslant w1$, there exists a least s for which there exists a w such that $\sigma \in OP_{w,s}$ and $w0 \leqslant w$.

Definition B.4. $SIG=(S, \leqslant, OP)$ is *locally upward/downward filtered* iff each connected component of $(S, \leqslant)$ is locally upward/downward filtered.

Definition B.5. A poset $(S, \leqslant)$ is *locally upward filtered* iff

$$\forall s,s' \in S \bullet \exists s'' \in S \bullet s,s' \leqslant s''.$$

Definition B.6. A poset $(S, \leqslant)$ is *locally downward filtered* iff

$$\forall s,s' \in S \bullet \exists s'' \in S \bullet s'' \leqslant s,s'.$$

Definition B.7. $SIG=(S, \leqslant, OP)$ is *coherent* iff SIG is regular and locally upward filtered.

B.2. Algebras and homomorphisms

Definition B.8. Given an order-sorted signature $SIG=(S, \leqslant, OP)$. An *order-sorted SIG-algebra*, A, is a many-sorted (S, OP)-algebra satisfying the following *monotonicity*

conditions
(i) $A_s \subseteq A_{s'}$, if $s \leqslant s'$,
(ii) $\forall a \in A_{w1} \bullet \sigma^A_{w1,s1}(a) = \sigma^A_{w2,s2}(a)$, if $\sigma \in OP_{w1,s1} \cap OP_{w2,s2}$ and $w1 \leqslant w2$.

Definition B.9. Given an order-sorted signature $SIG = (S, \leqslant, OP)$, and two SIG-algebras A and B. An *order-sorted SIG-homomorphism*, $h : A \mapsto B$, from A to B is a many-sorted (S, OP)-homomorphism from A to B satisfying the following *restriction condition*:

$$\forall \in A_s \bullet h_s(a) = h_{s'}(a), \quad \text{if } s \leqslant s'.$$

Notation. The category of order-sorted SIG-algebras and SIG-homomorphisms is denoted $OSAlg(SIG)$.

B.3. Terms and axioms

Definition B.10. Given an order-sorted signature $SIG = (S, \leqslant, OP)$.
Let $X = (X_s)_{s \in S}$ be a S-sorted set of *variables*. The sets, $T_s(SIG, X), s \in S$, of *SIG-terms* of sort s with variables in X, are inductively defined by the following rules:
(i) $x \in T_s(SIG, X)$, if $x \in X_s$,
(ii) $\sigma \in T_s(SIG, X)$, if $\sigma \in OP_s$,
(iii) $\sigma(t1, \ldots, tn) \in T_s(SIG, X)$, if $\sigma \in OP_{s1 \ldots sn, s}$ and $ti \in T_{si}(SIG, X)$ for $i = 1, \ldots, n$,
(iv) $t \in T_s(SIG, X)$, if $t \in T_{s'}(SIG, X)$ and $s' \leqslant s$.

Notation. We write $T_w(SIG, X)$ for $T_{s1}(SIG, X) \times \cdots \times T_{sn}(SIG, X)$, when $w = s1 \ldots sn$, and we write $\sigma(t)$ for $\sigma(t1, \ldots, tn)$, when $t = (t1, \ldots, tn)$.

Fact B.11. *A SIG-term may have several sorts, due to the last rule. For a regular signature SIG, any term t has a least sort, which we denote LS(t).*

Definition B.12. Given an order-sorted signature $SIG = (S, \leqslant, OP)$. The *SIG-term algebra* $T(SIG, X)$ is defined as follows:
(i) carriers: $T(SIG, X)_s = T_s(SIG, X)$,
(ii) constants: $\sigma_s^{T(SIG,X)} = \sigma$,
(iii) functions: $\forall t \in T(SIG, X)_w \bullet \sigma^{T(SIG,X)}_{w,s}(t) = \sigma(t)$.

Notation. We write $T(SIG)$ for $T(SIG, X)$, if X is empty. We write $T(X)$ for $T(SIG, X)$, and T for $T(SIG)$, when this does not give rise to confusion.

Theorem B.13. *Given a regular order-sorted signature SIG. Then $T(SIG)$ is initial in $OSAlg(SIG)$ and $T(SIG, X)$ is free over X in $OSAlg(SIG)$.*

Definition B.14. Given a regular order-sorted signature $SIG=(S,\leqslant,OP)$. An *order-sorted SIG-axiom* is a conditional equation of the form

$$(\forall X)t=t' \quad \textbf{if} \bigwedge_{i\in I}(t_i=t_i'),$$

where I is a finite or infinite set of indices; $t,t',t_i,t_i'\in T(SIG,X)$ for $i\in I$; $LS(t)$ and $LS(t')$ belong to the same connected component in $(S,\leqslant)$, and for each $i\in I$, $LS(t_i)$ and $LS(t_i')$ belong to the same connected component in $(S,\leqslant)$.

Notation. If I is empty, we just write $(\forall X)t=t'$. When the variable set X of an axiom can be deduced from the context, we allow the quantification to be omitted.

Definition B.15. Given a regular order-sorted signature $SIG=(S,\leqslant,OP)$, and a family X of SIG-variables. An *assignment*, $a:X\mapsto A$, is an S-sorted family $(a_s)_{s\in S}$ of functions $a_s:X_s\mapsto A_s$. The natural extension of a is the unique SIG-homomorphism, $a^*:T(SIG,X)\mapsto A$, for which $a^*(x)=x$ for $x\in X_s$.

Definition B.16. Given a regular order-sorted signature SIG. A SIG-algebra A *satisfies* a SIG-axiom, $(\forall X)t=t'$ **if** $\bigwedge_{i\in I}(t_i=t_i')$, if for every assignment $a:X\mapsto A$ for which $a^*_{LS(t_i)}(t_i)=a^*_{LS(t_i')}(t_i')$ for every $i\in I$, it holds that $a^*_{LS(t)}(t)=a^*_{LS(t')}(t')$.

Definition B.17. An order-sorted *specification* $SPEC$ is a pair (SIG,E) consisting of a regular order-sorted signature SIG and a set of order-sorted axioms E. A *SPEC-algebra* is a SIG-algebra satisfying all the axioms in E, and a *SPEC-homomorphism* is a SIG-homomorphism between $SPEC$-algebras.

Notation. The category of order-sorted $SPEC$-algebras and $SPEC$-homomorphisms is denoted $OSAlg(SPEC)$. The category of reachable $SPEC$-algebras and homomorphisms between these is denoted $ROSAlg(SPEC)$.

For a definition of the quotient term algebras $T(SPEC,X)$ and $T(SPEC)$, see [4].

Theorem B.18. *Given an order-sorted specification SPEC with coherent signature. Then $T(SPEC)$ is initial in $OSAlg(SPEC)$ and $T(SPEC,X)$ is free over X in $Alg(SPEC)$.*

Appendix C. First-order coercion algebra

In [17, 8] two notions of coercion algebras have been designed using category theory and set theory, respectively. In this section we develop our own notion of coercion algebra, which differs from [8, 17] by certain details concerning requirements to signatures and algebras, and is more close to the exposition for order-sorted algebra given in [4].

C.1. Signatures

Definition C.1. A coercion signature is a triple $(S, \leqslant, OP)$ such that

(i) (S, OP) is a many-sorted signature,

(ii) $(S, \leqslant)$ is a pre-order (i.e. reflexive and transitive),

(iii) the following *monotonicity condition* is satisfied: if $\sigma \in OP_{w1,s1} \cap OP_{w2,s2}$ and $w1 \leqslant w2$ then $s1 \leqslant s2$.

If we compare coercion signatures with order-sorted signatures the only difference is that the ordering need not be anti-symmetric.

Definition C.2. A coercion signature $(S, \leqslant, OP)$ is *pre-regular* iff for any $\sigma \in OP_{w1,s1}$ and $w0 \leqslant w1$, there exists a least sort s for which there exists a w such that $\sigma \in OP_{w,s}$ and $w0 \leqslant w$. Such a sort s is called *the least sort of* σ *over* $w0$, and is denoted $LS(\sigma, w0)$.

C.2. Algebras and homomorphisms

Definition C.3. Given a pre-regular coercion signature $CSIG = (S, \leqslant, OP)$. A *coercion CSIG-algebra*, A, consists of

(i) a many-sorted (S, OP)-algebra A,

(ii) coercion functions $c^A_{s,s'} : A_s \mapsto A_{s'}$, for each $s \leqslant s'$, satisfying the following conditions:

(a) *reflexivity*: $\forall a \in A_s \bullet c^A_{s,s}(a) = a$, for $s \in S$,

(b) *transitivity*: $\forall a \in A_s \bullet c^A_{s',s''}(c^A_{s,s'}(a)) = c^A_{s,s''}(a)$, for $s \leqslant s' \leqslant s''$ in S,

(c) *monotonicity*: $\forall a \in A_{w0} \bullet \sigma^A_{w1,s1}(c^A_{w0,w1}(a)) = c^A_{s2,s1}(\sigma^A_{w2,s2}(c^A_{w0,w2}(a)))$ for $\sigma \in OP_{w1,s1} \cap OP_{w2,s2} \wedge w0 \leqslant w1 \wedge w0 \leqslant w2 \wedge s2 = LS(\sigma, w0)$.

Notation. We write $c_{w,w'}(a)$ for $(c_{s1,s1'}(a1), \ldots, c_{sn,sn'}(an))$, when $w = s1 \ldots sn$, $w' = s1' \ldots sn'$ and $a = (a1, \ldots, an)$.

Definition C.4. Given a pre-regular coercion signature $CSIG = (S, \leqslant, OP)$, and two $CSIG$-algebras A and B. A *coercion CSIG-homomorphism*, $h : A \mapsto B$, from A to B is a many-sorted (S, OP)-homomorphism from A to B satisfying the following *restriction condition*:

$$\forall a \in A_s \bullet h_{s'}(c^A_{s,s'}(a)) = c^B_{s,s'}(h_s(a)), \quad \text{if } s \leqslant s'.$$

Notation. The $CSIG$-algebras and $CSIG$-homomorphisms form a category denoted $CAlg(CSIG)$.

C.3. Terms

Coercion $CSIG$-terms are defined exactly as order-sorted terms:

Definition C.5. Given a coercion signature $CSIG=(S,\leqslant,OP)$. Let $X=(X_s)_{s\in S}$ be a S-sorted set of variables. The sets, $T_s(CSIG,X)$, $s\in S$, of *CSIG-terms* of sort s with variables in X, are inductively defined by the following rules:

(i) $x\in T_s(CSIG,X)$, if $x\in X_s$,
(ii) $\sigma\in T_s(CSIG,X)$, if $\sigma\in OP_s$,
(iii) $\sigma(t)\in T_s(CSIG,X)$, if $\sigma\in OP_{w,s}$ and $t\in T_w(CSIG,X), w\neq\lambda$,
(iv) $t\in T_s(CSIG,X)$, if $t\in T_{s'}(CSIG,X)$ and $s'\leqslant s$.

Fact C.6. *A CSIG-term may have several sorts, due to the last rule. For a pre-regular signature CSIG, any term t has a least sort, LS(t).*

Definition C.7. Given a coercion signature $CSIG=(S,\leqslant,OP)$. The coercion *CSIG-term algebra* $T(CSIG,X)$ is defined as follows:

(i) carriers: $T(CSIG,X)_s=T_s(CSIG,X)$,
(ii) constants: $\sigma_s^{T(CSIG,X)}=\sigma$,
(iii) functions: $\forall t\in T(CSIG,X)_w \bullet \sigma_{w,s}^{T(CSIG,X)}(t)=\sigma(t)$,
(iv) coercion functions: $\forall t\in T(CSIG,X)_s \bullet c_{s,s'}^{T(CSIG,X)}(t)=t$.

Notation. We write $T(CSIG)$ for $T(CSIG,X)$, if X is empty. We write $T(X)$ for $T(CSIG,X)$, and T for $T(CSIG)$, when this does not give rise to confusion.

Theorem C.8. *Given a pre-regular coercion signature CSIG. Then T(CSIG) is initial in CAlg(CSIG).*

Proof. First we define a homomorphism $eval^A: T\mapsto A$, and then we prove that it is unique.

(i) We define $eval_s^A(t)$ by induction over the depth of t:

(a) $t=\sigma$: Let s be the unique sort for which $\sigma\in OP_s$, i.e. $s=LS(t)$. Then we define

$$eval_{s'}^A(\sigma)=c_{s,s'}^A(\sigma^A) \quad \text{for any } s' \text{ with } s\leqslant s'.$$

This especially means

$$eval_s^A(\sigma)=\sigma^A.$$

(b) $t=\sigma(t')$: Let $s=LS(t)$ (exists by pre-regularity), and w one of those arities for which $\sigma\in OP_{w,s}$. By induction hypothesis $eval_w^A(t')$ is defined. Then we define

$$eval_{s'}^A(\sigma(t'))=c_{s,s'}^A(\sigma_{w,s}^A(eval_w^A(t'))) \quad \text{for any } s' \text{ with } s\leqslant s'$$

This especially means

$$eval_s^A(\sigma(t'))=\sigma_{w,s}^A(eval_w^A(t'))).$$

Note, that the definition does not depend on the choice of the arity w, since $\sigma_{w',s}^A(eval_{w'}^A(t'))=\sigma_{w,s}^A(eval_w^A(t'))$ for any other w' for which $\sigma\in OP_{w',s}$.

(ii) $eval^A$ is a homomorphism:

(a) The restriction condition for constants $t = \sigma$: Let $s = LS(t)$. For any $s \leqslant s1 \leqslant s2$ we have

$$\begin{aligned}
&eval^A_{s2}(c^T_{s1,s2}(\sigma)) \\
&\quad = eval^A_{s2}(\sigma) \\
&\quad = c^A_{s,s2}(\sigma^A) \\
&\quad = c^A_{s1,s2}(c^A_{s,s1}(\sigma^A)) \\
&\quad = c^A_{s1,s2}(eval^A_{s1}(\sigma)).
\end{aligned}$$

(b) The restriction condition for non-constants $t = \sigma(t')$: Let $s = LS(t)$ and w one of those arities for which $\sigma \in OP_{w,s}$. For any $s \leqslant s1 \leqslant s2$ we have

$$\begin{aligned}
&eval^A_{s2}(c^T_{s1,s2}(\sigma(t'))) \\
&\quad = eval^A_{s2}(\sigma(t')) \\
&\quad = c^A_{s,s2}(\sigma^A_{w,s}(eval^A_w(t'))) \\
&\quad = c^A_{s,s2}(eval^A_s(\sigma(t'))) \\
&\quad = c^A_{s1,s2}(c^A_{s,s1}((eval^A_s(\sigma(t'))))) \\
&\quad = c^A_{s1,s2}(c^A_{s,s1}(\sigma^A_{w,s}(eval^A_w(t')))) \\
&\quad = c^A_{s1,s2}(eval^A_{s1}(c^T_{s,s1}(\sigma(t')))) \\
&\quad = c^A_{s1,s2}(eval^A_{s1}(\sigma(t'))).
\end{aligned}$$

(c) Homomorphism condition for constant symbols $\sigma \in OP_s$:

$$eval^A_s(\sigma^T) = eval^A_s(\sigma) = \sigma^A$$

(d) Homomorphism condition for non-constant symbols $\sigma \in OP_{w',s'}$: Let $t' \in T_{w'}$, $w0 = LS(t')$ and $s = LS(\sigma(t'))$. (Then $w0 \leqslant w'$ and $s \leqslant s'$.)

$$\begin{aligned}
&eval^A_{s'}(\sigma^T_{w',s'}(t')) \\
&\quad = eval^A_{s'}(\sigma(t')) && \text{(by definition of term algebra)} \\
&\quad = c^A_{s,s'}(\sigma^A_{w,s}(eval^A_w(t'))) && \text{(by definition of eval)} \\
&\quad = c^A_{s,s'}(\sigma^A_{w,s}(eval^A_w(c^T_{w0,w}(t')))) && \text{(by definition of term algebra)} \\
&\quad = c^A_{s,s'}(\sigma^A_{w,s}(c^A_{w0,w}(eval^A_{w0}(t')))) && \text{(by restriction condition)} \\
&\quad = \sigma^A_{w',s'}(c^A_{w0,w'}(eval^A_{w0}(t'))) && \text{(by monotonicity condition)} \\
&\quad = \sigma^A_{w',s'}(eval^A_{w'}(c^T_{w0,w'}(t'))) && \text{(by restriction condition)} \\
&\quad = \sigma^A_{w',s'}(eval^A_{w'}(t')) && \text{(by definition of term algebra).}
\end{aligned}$$

(iii) $eval^A$ is unique: Assume, that there is another homomorphism $h^A : T \mapsto A$. We then prove that $h^A_s(t) = eval^A_s(t)$ by induction over the depth of t:

(a) $t = \sigma$: obvious.

(b) $t = \sigma(t')$, $\sigma \in OP_{w',s'}$: By the induction hypothesis we have $h^A_{w'}(t') = eval^A_{w'}(t')$. Let $w0 = LS(t')$, $s = LS(\sigma(t'))$, and w be such that $w0 \leqslant w$ and $\sigma \in OP_{w,s}$ (Then $w0 \leqslant w'$ and $s \leqslant s'$.):

$$
\begin{aligned}
& h^A_{s'}(\sigma(t')) \\
&= h^A_{s'}(\sigma^T_{w',s'}(t')) && \text{(by definition of term algebra)} \\
&= \sigma^A_{w',s'}(h^A_{w'}(t')) && \text{(by homomorphism condition)} \\
&= \sigma^A_{w',s'}(eval^A_{w'}(t')) && \text{(by induction hypothesis)} \\
&= \sigma^A_{w',s'}(eval^A_{w'}(c^T_{w0,w'}(t'))) && \text{(by definition of term algebra)} \\
&= \sigma^A_{w',s'}(c^A_{w0,w'}(eval^A_{w0}(t'))) && \text{(by restriction condition)} \\
&= c^A_{s,s'}(\sigma^A_{w,s}(c^A_{w0,w}(eval^A_{w0}(t')))) && \text{(by monotonicity condition)} \\
&= c^A_{s,s'}(\sigma^A_{w,s}(eval^A_{w}(c^T_{w0,w}(t')))) && \text{(by restriction condition)} \\
&= c^A_{s,s'}(\sigma^A_{w,s}(eval^A_{w}(t'))) && \text{(by definition of term algebra)} \\
&= eval^A_{s'}(\sigma(t')) && \text{(by definition of eval).} \quad \square
\end{aligned}
$$

Theorem C.9. *Given a pre-regular coercion signature CSIG. Then $T(CSIG,X)$ is free over X in $CAlg(CSIG)$.*

Proof. The proof is entirely analogous to that of the corresponding theorem for order-sorted algerba in [4]. □

C.4. Axioms

In coercion algebra we need a more refined notion of axioms than in (modified) order-sorted algebra for the following reasons. In modified order-sorted algebra, the evaluation of a term, t, in an algebra, A, does not depend on over which of its sorts the evaluation is done. This is due to the fact that $a^*_s(t) = a^*_{LS(t)}(t)$ for $LS(t) \leqslant s$. Therefore, over which sort the terms, $t1$ and $t2$, of an equation, $t1 = t2$, is evaluated is not important. However, in coercion algebra this is not the case due to the fact that $a^*_s(t) = c^A_{LS(t),s}(a^*_{LS(t)}(t))$ for $LS(t) \leqslant s$, and the fact that coercions need not be injective. Hence, the evaluation of $t1$ and $t2$ may give the same value over one upper bound of $LS(t1)$ and $LS(t2)$, but different values for another upper bound. Therefore, in coercion algebra we need a more refined notion of equations, where we can indicate over which sort the equality should hold. We do this by giving the sort as a subscript of the equality operator.

Notation. Given a pre-order $(S, \leqslant)$. For $s1, s2 \in S$, $UB(s1, s2)$ denotes the set of upper bounds of $s1$ and $s2$ in $(S, \leqslant)$.

Definition C.10. Given a pre-regular coercion signature $CSIG=(S,\leqslant,OP)$. A *coercion CSIG-axiom* is a conditional equation of the form

$$(\forall X)\, t =_{s0} t' \quad \textbf{if} \bigwedge_{i\in I} (t_i =_{si} t_i'),$$

where I is a finite or infinite set of indices; $t,t',t_i,t_i' \in T(CSIG,X)$ for all $i\in I$; $s0 \in UB(LS(t),LS(t'))$ and $si \in UB(LS(t_i),LS(t_i'))$ for all $i\in I$.

Notation. We write $t=t'$ as a shorthand for $\bigwedge_{s\in UB(LS(t),\,LS(t'))} t =_s t'$.

Definition C.11. Given a pre-regular coercion signature $CSIG=(S,\leqslant,OP)$, and a family X of $CSIG$-variables. An *assignment*, $a: X \mapsto A$, is an S-sorted family $(a_s)_{s\in S}$ of functions $a_s: X_s \mapsto A_s$. The natural extension of a is the unique $CSIG$-homomorphism, $a^*: T(CSIG,X) \mapsto A$, for which $a^*(x)=x$ for $x\in X_s$.

Definition C.12. Given a pre-regular coercion signature $CSIG$. A $CSIG$-algebra A *satisfies* a $CSIG$-axiom, $(\forall X)\ t =_{s0} t'$ **if** $\bigwedge_{i\in I} (t_i =_{s_i} t_i')$, if for every assignment $a: X \mapsto A$ for which $a^*_{si}(t_i) = a^*_{si}(t_i')$ for every $i\in I$, it holds that $a^*_{s0}(t) = a^*_{s0}(t')$.

Fact C.13. *If a CSIG-algebra, A, satisifes* $t =_s t'$ *and* $s \leqslant s'$, *then A also satisfies* $t =_{s'} t'$, *but not vice versa.*

Definition C.14. A coercion *specification CSPEC* is a pair $(CSIG,E)$ consisting of a pre-regular coercion signature $CSIG$ and a set of coercion axioms E. A *CSPEC-algebra* is a $CSIG$-algebra satisfying all the axioms in E, and a *CSPEC-homomorphism* is a $CSIG$-homomorphism between $CSPEC$-algebras.

Notation. The $CSPEC$-algebras and $SPEC$-homomorphisms form a category denoted $CAlg(CSPEC)$.

Theorem C.15. *There exists an initial algebra in* $CAlg(CSPEC)$.

Proof. Follows from the reduction Theorem C.20 and the initiality theorem for many-sorted algebra. □

C.5. Generalisation of modified order-sorted algebra

Coercion algebra is a generalisation of modified order-sorted algebra in the sense that any order-sorted signature $(S,\leqslant,OP)$ is also a coercion signature, and any modified order-sorted $(S,\leqslant,OP)$-algebra can be considered as a coercion $(S,\leqslant,OP)$-algebra with $c^A_{s,s'}(a)=a$ for $s\leqslant s'$. Furthermore, any modified order-sorted $(S,\leqslant,OP)$-axiom is also a coercion $(S,\leqslant,OP)$-axiom which is satisfied by the same algebras.

C.6. Reduction to many-sorted algebra

For any pre-regular coercion-signature $CSIG = (S, \leqslant, OP)$, we can give a many-sorted specification, $CSIG^{\prec}$, such that any coercion $CSIG$-algebra can be considered as a many-sorted $CSIG^{\prec}$-algebra and vice versa.

Definition C.16. The many-sorted specification *associated* with a coercion signature CSIG is

$$CSIG^{\prec} = (CSIG^{\#}, coercion\_ax(CSIG))$$

where $CSIG^{\#} = (S, disambiguated\_op(OP) \cup coercion\_op(S, \leqslant))$, $disambiguated\_op(OP) = \{\sigma_{w,s} : w \mapsto s \mid \sigma : w \mapsto s \in OP\}$, $coercion\_op(S, \leqslant) = \{c_{s,s'} : s \mapsto s' \mid s \leqslant s' \text{ in } S\}$, and $coercion\_ax(CSIG)$ consists of the following axioms:

(i) (reflexivity) $c_{s,s}(x) = x$, for $s \in S$,

(ii) (transitivity) $c_{s',s''}(c_{s,s'}(x)) = c_{s,s''}(x)$, for $s \leqslant s' \leqslant s''$ in S,

(iii) (monotonicity) $\sigma_{w1,s1}(c_{w0,w1}(x)) = c_{s2,s1}(\sigma_{w2,s2}(c_{w0,w2}(x)))$, for $\sigma \in OP_{w1,s1} \cap OP_{w2,s2} \wedge w0 \leqslant w1 \wedge w0 \leqslant w2 \wedge s2 = LS(\sigma, w0)$.

Any coercion $CSIG$-algebra A can be considered as a many-sorted $CSIG^{\prec}$-algebra $A^{\#}$ by letting

(i) $(A^{\#})_s = A_s$, for $s \in S$,

(ii) $(\sigma_{w,s})^{A^{\#}}_{w,s} = \sigma^{A}_{w,s}$, for $\sigma \in OP_{w,s}$,

(iii) $(c_{s,s'})^{A^{\#}}_{s,s'} = c^{A}_{s,s'}$, for $s \leqslant s'$.

Similarly, any many-sorted $CSIG^{\prec}$-algebra B can be considered as a coercion $CSIG$-algebra $B^{\bullet}$ by letting

(i) $(B^{\bullet})_s = B_s$, for $s \in S$,

(ii) $\sigma^{B^{\bullet}}_{w,s} = (\sigma_{w,s})^{B}_{w,s}$, for $\sigma \in OP_{w,s}$,

(iii) $c^{B^{\bullet}}_{s,s'} = (c_{s,s'})^{B}_{s,s'}$, for $s \leqslant s'$.

The two constructions above extends in a natural way to homomorphisms by letting $(h^{\#})_s(x) = h_s(x)$ for $s \in S$, and $(h^{\bullet})_s(x) = h_s(x)$ for $s \in S$. In this way we have defined two functors, $(\_)^{\#} : CAlg(CSIG) \to Alg(CSIG^{\prec})$ and $(\_)^{\bullet} : Alg(CSIG^{\prec}) \to CAlg(CSIG)$.

We can now formulate the reduction theorem for $CAlg(CSIG)$:

Theorem C.17. *$CAlg(CSIG) = Alg(CSIG^{\prec})$, in the sense that $(\_)^{\#\bullet}$ and $(\_)^{\bullet\#}$ are identity functors on $CAlg(CSIG)$ and $Alg(CSIG^{\prec})$, respectively.*

Proof. Obvious from the definitions of the two functors $(\_)^{\#}$ and $(\_)^{\bullet}$. □

In order to give a reduction theorem for $CAlg(CSPEC)$, we first need to define a translation, *parses*, of $CSIG$-equations to $CSIG^{\prec}$-equations, and a translation, *unparse*, of $CSIG^{\prec}$-equations to $CSIG$-equations.

unparse(e) is obtained from e by removing any application of coercion functions, and by replacing occurences of operation symbols $\sigma_{w,s}$ with the corresponding ambiguous operation symbols σ:

$$\begin{aligned} &\mathit{unparse}((\forall X)\ t = t' \quad \textbf{if} \bigwedge_{i\in I} (t_i = t_i')) \\ &\quad = ((\forall X)\, h_s(t) =_s h_s(t') \quad \textbf{if} \bigwedge_{i\in I} (h_{si}(t_i) =_{s_i} h_{si}(t_i'))), \end{aligned}$$

where $h: T(CSIG^{\#}, X) \to T(CSIG, X)^{\#}$ is the unique homomorphism extending the inclusion of X in $T(CSIG, X)^{\#}$, s is the sort of t and t', and s_i is the sort of t_i and t_i'.

The set of possible parses of a *CSIG*-equation is defined as follows:

$$\mathit{parses}(e) = \{e' \mid \mathit{unparse}(e') = e\}.$$

Lemma C.18. *For any CSIG-algebra A, CSIG$^{\prec}$-algebra B, CSIG-axiom e and CSIG$^{\#}$-axiom e', the following holds:*
(i) *A satisfies e iff $A^{\#}$ satisfies parses(e),*
(ii) *B satisfies e' iff $B^{\bullet}$ satisfies unparse(e').*

Proof. The proof is analogous to the proof of the corresponding result for order-sorted algebra in [4]. □

For any pre-regular coercion specification $CSPEC = (CSIG, E)$, we can give a many-sorted specification $CSPEC^{\prec}$, such that any coercion *CSPEC*-algebra can be considered as a many-sorted $CSPEC^{\prec}$-algebra and vice versa.

Definition C.19. The many-sorted specification *associated* with a coercion specification CSPEC is

$$CSPEC^{\prec} = CSIG^{\prec} \cup \mathit{parses}(E),$$

where $\mathit{parses}(E) = \{\mathit{parses}(e) \mid e \in E\}$.

We can now formulate the reduction theorem for *CAlg*(*CSPEC*):

Theorem C.20. *CAlg(CSPEC) = Alg($CSPEC^{\prec}$), in the sense that $(\_)^{\#\bullet}$ and $(\_)^{\bullet\#}$ are the identity functors on CAlg(CSPEC) and Alg($CSPEC^{\prec}$), respectively.*

Proof. From Lemma C.18 it follows that the previously defined functors restrict to functors $(\_)^{\#}: CAlg(CSPEC) \to Alg(CSPEC^{\prec})$ and $(\_)^{\bullet}: Alg(CSPEC^{\prec}) \to CAlg(CSPEC)$, respectively. □

Appendix D. First-order generalised algebra

Definition D.1. A *generalised signature* is a four-tuple $(S, \leqslant_1, \leqslant_2, OP)$ such that
(i) $(S, \leqslant_1, OP)$ is a pre-regular order-sorted signature,
(ii) $(S, \leqslant_2, OP)$ is a pre-regular coercion signature,
(iii) $\leqslant_1$ is a subrelation of $\leqslant_2$.

Definition D.2. Given a generalised signature $GSIG = (S, \leqslant_1, \leqslant_2, OP)$. A *generalised GSIG-axioms* is a coercion $(S, \leqslant_2, OP)$-axiom, as defined in Appendix C.

Definition D.3. Given a generalised signature $GSIG = (S, \leqslant_1, \leqslant_2, OP)$. A *generalised GSIG-algebra*, A, is a coercion $(S, \leqslant_2, OP)$-algebra with the additional property that the following *inclusion condition* is satisfied:

$$A_s \subseteq A_{s'} \text{ and } \forall a \in A_s \bullet c^A_{s,s'}(a) = a \quad \text{for } s \leqslant_1 s'$$

As generalised *GSIG*-algebras are just special coercion $(S, \leqslant_2, OP)$-algebras, we can define *GSIG*-homomorphisms as follows:

Definition D.4. Given a generalised signature $GSIG = (S, \leqslant_1, \leqslant_2, OP)$. A *generalised GSIG-homomorphism* is a coercion $(S, \leqslant_2, OP)$-homomorphism between *GSIG*-algebras.

Notation. The *GSIG*-algebras and *GSIG*-homomorphisms form a category denoted $GAlg(GSIG)$.

Theorem D.5. *There exists an initial algebra in* $GAlg(GSIG)$.

Proof. Follows from the reduction Theorem D.10 and the initiality Theorem C.8 for coercion algebra. □

As generalised *GSIG*-algebras and *GSIG*-axioms are just special coercion $(S, \leqslant_2, OP)$-algebras and $(S, \leqslant_2, OP)$-axioms, respectively, we define *GSIG-satisfaction* as follows:

Definition D.6. Given a generalised signature $GSIG = (S, \leqslant_1, \leqslant_2, OP)$. A *GSIG*-algebra A *satisfies* a *GSIG*-axiom e, if A coercion $(S, \leqslant_2, OP)$-satisfies e.

Definition D.7. A generalised *specification GSPEC* is a pair $(GSIG, E)$ consisting of a generalised signature *GSIG* and a set of generalised *GSIG*-axioms E. A *GSPEC-algebra* is a *GSIG*-algebra satisfying all the axioms in E, and a *GSPEC-homomorphism* is a *GSIG*-homomorphism between *GSPEC*-algebras.

Notation. The *GSPEC*-algebras and *GSPEC*-homomorphisms form a category denoted $GAlg(GSPEC)$. The category of reachable *GSPEC*-algebras and *GSPEC*-homomorphisms between these is denoted $RGAlg(GSPEC)$.

Theorem D.8. *There exists an initial algebra in* $GAlg(GSPEC)$.

Proof. Follows from the Reduction Theorem D.12 and the initiality Theorem C.15 for coercion algebra. □

D.1. Generalisation of coercion and modified order-sorted algebra

Generalised algebras are generalisations of coercion algebras in the sense that any coercion $(S, \leqslant, OP)$-algebra can be considered as a generalised $(S, \leqslant_{\min}, \leqslant, OP)$-algebra, where $\leqslant_{\min}$ is the least ordering relation (i.e. $s1 \leqslant_{\min} s2$, only if $s1 = s2$). Furthermore, any coercion $(S, \leqslant, OP)$-axiom is also a generalised $(S, \leqslant_{\min}, \leqslant, OP)$-axiom.

Generalised algebras are generalisations of modified order-sorted algebras in the sense that any modified order-sorted $(S, \leqslant, OP)$-algebra can be considered as a generalised $(S, \leqslant, \leqslant, OP)$-algebra. Furthermore, any modified order-sorted $(S, \leqslant, OP)$-axiom is also a generalised $(S, \leqslant, \leqslant, OP)$-axiom.

D.2. Reduction of coercion algebra

For any generalised signature $GSIG = (S, \leqslant_1, \leqslant_2, OP)$, we can give a coercion specification $GSIG^{\prec:}$, such that the $GSIG^{\prec:}$-algebras are isomorphic to the *GSIG*-algebras:

Definition D.9. The coercion specification associated with a generalised signature $GSIG = (S, \leqslant_1, \leqslant_2, OP)$ is

$$GSIG^{\prec:} = ((S, \leqslant_2, OP), injectivity\_ax(S, \leqslant_1)),$$

where $injectivity\_ax(S, \leqslant_1) = \{x =_s y \textbf{ if } x =_{s'} y \mid s \leqslant_1 s'\}$.

We can now formulate the reduction theorem for $GAlg(GSIG)$:

Theorem D.10. $GAlg(GSIG) \cong CAlg(GSIG^{\prec:})$, *in the sense that*

(i) $GAlg(GSIG) \subseteq CAlg(GSIG^{\prec:})$ ('$\subseteq$' *means 'is full subcategory of'*)

(ii) *there exists a functor* $(\_)^\bullet : CAlg(GSIG^{\prec:}) \to GAlg(GSIG)$ *for which there is an natural isomorphism* $j : id \to (\_)^{\bullet\#}$, *where id is the identity functor on CAlg* $(GSIG^{\prec:})$ *and* $(\_)^\# : GAlg(GSIG) \to CAlg(GSIG^{\prec:})$ *is the inclusion functor, i.e.*

(a) *for each* $GSIG^{\prec:}$*-algebra B there is an isomorphism* $j^B : B \to B^\bullet$. *in CAlg* $(GSIG^{\prec:})$

(b) *for each* $GSIG^{\prec:}$*-homomorphism,* $h : B1 \to B2$, *and* $s \in S$

$$h_s^\bullet(j_s^{B1}(b)) = j_s^{B2}(h_s(b))).$$

Proof. (i) The first part is obvious.

(ii) For the second part, the functor $(\_)^\bullet$ is defined as follows: For any $GSIG^{\prec:}$-algebra B, let $(B_{\mathrm{col}}, (j_s^B : B_s \to B_{\mathrm{col}})_{s \in S})$ be the colimit of the diagram consisting of the carrier sets B_s for $s \in S$, and the coercion functions $c_{s,s'}^B : B_s \to B_{s'}$ for $s \leqslant_1 s'$. As the coercion functions are injective, also the colimit functions, j_s^B, are injective. For $s \in S$ we now define

$$(B^\bullet)_s = j_s^B(B_s).$$

Hence, the functions $j_s^B : B_s \to B_{s'}$ are bijections, and we can then define $\sigma_{w,s}^{B^\bullet}$ for $\sigma \in OP_{w,s}$ by the following equation:

$$\sigma_{w,s}^{B^\bullet}(j_w^B(x)) = j_s^B(\sigma_{w,s}^B(x)).$$

It is easily checked that $B^\bullet$ is a $GSIG$-algebra.

For any $GSIG^{\prec:}$-homomorphism $h : B1 \to B2$, we define the homomorphism $h^\bullet : B1^\bullet \to B2^\bullet$ by

$$(h^\bullet)_s(j_s^{B1}(b)) = j_s^{B2}(h_s(b)))$$

for $s \in S$. It is easily checked that $h^\bullet$ is a $GSIG$-homomorphism.

$j^B = (j_s^B)_{s \in S} : B \to B^\bullet$ is an isomorphism as it is an homomorphism and the j_s^B functions are bijective.

Hence, the functor $(\_)^\bullet$ has the required properties. □

Lemma D.11. *For any $GSIG^{\prec:}$-algebra B and GSIG-axiom e, B satisfies e iff $B^\bullet$ satisfies e.*

Proof. Follows from the fact that equational satisfaction is closed under isomorphim and B is isomorphic to $B^\bullet$. □

We can now formulate the reduction theorem for $GAlg(GSIG, E)$:

Theorem D.12. $GAlg(GSIG, E) \cong CAlg(GSIG^{\prec:} \cup E)$.

Proof. Follows from Theorem D.10 and Lemma D.11. □

Acknowledgements

I would like to thank Professor Futatsugi for inviting me to Japan and for scientific discussions, and Professor Tarlecki and Professor Möller for scientific discussions.

References

[1] H. Ehrig, B. Mahr, Fundamentals of Algebraic Specification 1, Equations and Initial Semantics, EATCS Monographs on Theoretical Computer Science, vol. 6, Springer, Berlin, 1985.

[2] K. Futatsugi, J. Goguen, J. Jouannaud, J. Meseguer, Principles of OBJ2, in: Proc. POPL'85, Association for Computing Machinery, 1985, pp. 52–66.
[3] J. Goguen, Higher-order functions considered unnecessary for higher order programming, Technical Report SRI-CSL-88-1R, SRI Int., 1988.
[4] J. Goguen, J. Meseguer, Order-sorted algebra I: equational deduction for multiple inheritance, overloading, exceptions and partial operations, Theoret. Comput. Sci. 105(2) (1992) 217–273.
[5] J. Goguen, T. Winkler, J. Meseguer, K. Futatsugi, J. Jouannaud, Introducing OBJ, Tech. Report SRI-CSL-92-03, SRI Int., 1992.
[6] A.E. Haxthausen, Algebraic specification with higher-order functions, Tech. Report ETL TR-94-18, Electrotechnical Laboratory, 1994.
[7] A.E. Haxthausen, Order-sorted algebraic specifications with higher-order functions, in: Proc. AMAST'95, Lecture Notes in Computer Science, vol. 936, Springer, Berlin, 1995, pp. 133–151.
[8] H. Kreowski, Z. Qian, Relation-sorted algebraic specifications with built-in coercers: Basic notions and results, in: Proc. STACS'90, Lecture Notes in Computer Science, vol. 415, Springer, Berlin, 1990, pp. 165–175.
[9] N. Martí-Oliet, J. Meseguer, Inclusions and subtypes, Tech. Report SRI-CSL-90-16, SRI Int., 1990.
[10] K. Meinke, Universal algebra in higher types, Theoret. Comput. Sci. 100(2) (1992) 385–417.
[11] B. Möller, Algebraic specification with higher-order operations, in: Proc. IFIP TC2 Working Conf. on Program Specification and Transformation, North-Holland, Amsterdam, 1986, pp. 367–392.
[12] B. Möller, A. Tarlecki, M. Wirsing, Algebraic specification of reachable higher-order algebras, in: 5th Workshop on Specification of Abstract Datatypes, Lecture Notes in Computer Science, vol. 332, Springer, Berlin, 1988, pp. 154–169.
[13] B. Möller, A. Tarlecki, M. Wirsing, Algebraic specifications with built-in domain constructions, in: Proc. CAAP'88, Lecture Notes in Computer Science, vol. 299, Springer, Berlin, 1988, pp. 132–148.
[14] A. Poigné, On specifications, theories and models with higher types, Inform. Control 68 (1986) 1–46.
[15] Z. Qian, Higher-order order-sorted algebras, in: Proc. Algebraic and Logic Programming, Lecture Notes in Computer Science, vol. 463, Springer, Berlin, 1990, pp. 86–100.
[16] The RAISE Language Group, The RAISE Specification Language, The BCS Practitioners Series, Prentice-Hall, Englewood Cliffs, NJ, 1992.
[17] J. Reynolds, Using category theory to design implicit conversions and generic operators, in: Proc. Semantics-Directed Compiler Generation, Lecture Notes in Computer Science, vol. 94, Springer, Berlin, 1980, pp. 211–258.